**一、代码及运行截图：**

#include <stdio.h>

#include <stdlib.h>

#include <math.h>

int c=0;

void INPUT(int \*p,int \*q)

{

printf("输入两个整数:");

scanf("%d %d",p,q);

}

int Step1(int x)

{

c++;

printf("向左滑步\n");

return x+1;

}

int Step2(int x)

{

c++;

printf("向右滑步\n");

return x-1;

}

int Jump(int x)

{

c++;

printf("向右跳步\n");

return x/2;

}

SHIFT(a,b)

{

if (b%2!=0)

{

int c=(b+1)/2;

int d=(b-1)/2;

if (abs(c-a)>=b-a || abs(d-a)>=b-a)

{

while (a!=b)

{

b=Step2(b);

}

}

else if (abs(c-a)<abs(d-a))

{

b=Step1(b);

b=Jump(b);

SHIFT(a,b);

}

else if (abs(c-a)>abs(d-a))

{

b=Step2(b);

b=Jump(b);

SHIFT(a,b);

}

}

else

{

if (abs(b/2 - a) > abs(b - a))

{

if (b>a)

{

while(a!=b)

{

b=Step2(b);

return 0;

}

}

else if (b<a)

{

while(a!=b)

{

b=Step1(b);

return 0;

}

}

else

{

return 0;

}

}

else if (abs(b/2 - a) < abs(b - a))

{

b=Jump(b);

if (b>a)

SHIFT(a,b);

else

{

while(a!=b)

{

b=Step1(b);

}

}

}

else

{

b=Jump(b);

return 0;

}

}

}

void OUTPUT()

{

printf("Bob最少需要%d次才能捡到篮球.",c);

}

int main()

{

int N,K,x;

int \*p,\*q;

p=&N,q=&K;

INPUT(p,q);

x=K;

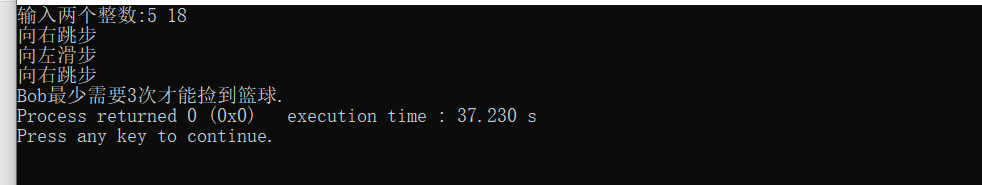
SHIFT(N,x);

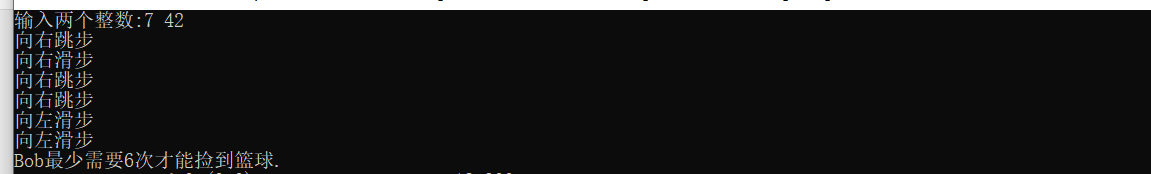
OUTPUT();

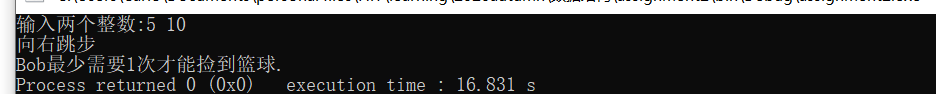
}











**二、我的算法：**

1.先对终点就行奇偶性判定

2.若是偶数，则让终点除以2，采用递归的方法对新的始点和终点继续运行

3.若是奇数，则判定终点加一除以二和终点减一除以二与始点的距离比较

4.若新终点和始点距离更短的话，则一步步滑步就可以完成

5.若新终点和始点距离更远的话，则继续判断和递归。

**三、使用的数据结构**

1.栈的递归应用

2.树的思考方法

**四、时间复杂性和空间复杂性分析**

1.时间复杂性:

每次递归都除以2，整体来说

2.空间复杂性：

未用到数组等结构，每递归一次压栈一次，空间复杂性整体来说